CS 61C Summer 2016
Guerrilla Section 2: MIPS

Question 1: What's that funky smell? Oh, it's Potpourri.

a) Decode the binary numbers into MIPS instructions with proper register names ($s0, $t0, etc.). If there are any memory addresses, represent them in hex.

Address	32-bit Binary Instruction	Type (R, I, J)	MIPS Instruction w/args
0xAFFFFFF8	0000 0001 0000 1000 0100 0000 0010 0110	R	xor $t0, $t0, $t0
0xAFFFFFFC	0001 0100 0000 1000 1111 1111 1111 1110	I	bne $0, $t0, -2
0xB0000000	0000 1000 0000 0000 0000 0000 0000 0001	J	j 0xB0000004
0xB0000004	...whatever...	...whatever...	ori $v0, $0, 0x61C
0xB0000008	...whatever...	...whatever...	jr $ra

b) A variant of the MIPS instruction set has been developed on the new 64-bit system for cloud computing. To emphasize the "big" in big data, the size of the "opcode" field has been increased by one bit, and the number of registers (whose width is now 64-bits) was quadrupled. Assume that instructions "expand" the rightmost field to fill all 64 bits. so here, the "funct" field would expand; in an I-type instructions, the Immediate would expand. You may use the boxes below as scratch space.

7	7	7	7	6	30
Opcode	Rs	Rt	Rd	Shamt	Funct

a) How many total instructions can we have? Please leave your answer as an expression involving powers of 2.

$$(2^7 - 1) \text{ non-R-type} + 2^{30} \text{ R-type}$$

b) What is the maximum amount of bytes we could change the PC by with a single branch instruction?
(Express your answer in IEC format, e.g., 32 Kibi, 16 Mebi, etc.)

Bits for Imm = 64 - 7*3 = 43, counting by instruction that's 8 bytes (3 free) = (2^43) * 8 = 2^46 bytes. Can branch forwards and backwards, range is 2^45 = 32 Tebi bytes for max branch.

Question 2 (Fa15 Q5) MIPS Sleuth.

```
1      mystery:    addiu $sp $sp _____-16_____
2                  _____sw $s0 0($sp)_____
3                  _____sw $s1 4($sp)_____
4                  _____sw $s2 8($sp)_____
5                  _____sw $ra 12($sp)_____
6                  _____
7                  addu $s0 $0 $0
8                  move $s1 $a0
9                  move $s2 $a1
10     loop:       srl $t0 $s0 2
11                 beq $t0 $s2 exit
12                 subu $a0 $s2 $t0
13                 jal random
14                 sll $v0 $v0 2
15                 addu $v0 $v0 $s0
16                 addu $t0 $s1 $s0
17                 addu $t1 $s1 $v0
18                 lw $t2 0($t0)
19                 lw $t3 0($t1)
20                 sw $t2 0($t1)
21                 sw $t3 0($t0)
22                 addiu $s0 $s0 4
23                 j loop
24     exit:       _____lw $s0 0($sp)_____
25                 _____lw $s1 4($sp)_____
26                 _____lw $s2 8($sp)_____
27                 _____lw $ra 12($sp)_____
28                 _____addiu $sp $sp 16_____
29                 _____jr $ra_____
30                 _____
```

1) Fill in the prologue and the epilogue of this MIPS function.
2) What operation does this function perform on an integer array? Assume that both the integer array and the length of the array are passed into the function.
   The function shuffles the integer array in place.
3) Would this function work as expected if a string was passed into the function instead? Write down the line numbers of all lines of MIPS code that must be changed (if any at all), so that the function works correctly on strings. Do not write down any extraneous line numbers
   10, 14, 18, 19, 20, 21, 22

Question 3: *"free at last, thank gosh we are free at last…"*

We wish to free a linked list of strings (example below) whose nodes are made up of this struct. Complete the code below; we have started you off with some filled in. You may use fewer lines, but do not add any.

```
FreeLL:     beq $a0, $0, NULL_CASE
            addiu $sp $sp -8
            sw $ra 4($sp)
            sw $a0 0($sp)
            lw $a0 4($a0)

            _____
            jal FreeLL
            lw $a0 0($sp)
            lw $a0 0($a0)
            jal free
            lw $a0 0($sp)

            _____
            jal free
            lw $ra 4($sp)
            addiu $sp $sp 8

            _____
NULL_CASE:      jr $ra
```

Question 5 (Fa15 Q7) After this, you're CALL done! (9 points)

Connect the definition with the name of the process that describes it.

    a) Compiler
    b) Assembler
    c) Linker
    d) Loader

__a__1) Outputs code that may still contain pseudoinstructions.
__d__2) Takes binaries stored on disk and places them in memory to run.
__b__3) Makes two passes over the code to solve the "forward reference" problem.
__b__4) Creates a symbol table.
__c__5) Combines multiple text and data segments.
__a__6) Generates assembly language code.
__b__7) Generates machine language code.
__b__8) Only allows generation of TAL.
__c__9) Only allows generation of binary machine code.